REMARKS

Status of the Claims

Claims 10-111, as amended, remain in the case and new claims 112-123 are added to the case. No new matter is added by this amendment.

Claims 10-111 have been rejected.

Claim Rejections under 35 USC § 102

Claims 10-111 are rejected under 35 U.S.C. § 102(e) as being anticipated by Nakano et al. (US 5,369,766).

Applicants' Response to Rejections under 35 USC § 102

The Applicants' remarks in the previous amendment of June 3, 2007 are incorporated herein by reference.

The Applicants' claimed invention allows the same application to run on different operating system platforms, because the Applicants have invented a common object-oriented interface implemented for each platform. The Applicants' claimed invention enables the same object-oriented application to run on the different combinations of operating system and hardware, because of the object-oriented interface is implemented for each combination. The Applicants' have amended their claims to emphasize that the interface is used by the same object-oriented application on the plurality of computer platforms to instantiate objects from the classes and invoke the object-oriented methods. The amended claims further emphasize that the program logic code on any one of the plurality of computer platforms is responsive to the object-oriented interface implemented on the one computer platform to provide native operating system services from the one computer platform.

The Office Action repeated the prior art rejection of March 9, 2007 over the Nakano reference and has added some additional reasons for the rejection. The Examiner's reasons listed in boldface type in the September 7, 2007 Office action, are three paragraphs beginning at the bottom of page 3 and continuing to the bottom of page 4. The Applicant will address each of the three paragraphs as follows.

A. The Examiner's first boldface paragraph:

The Examiner's first boldface paragraph: September 7, 2007 Office Action, Page 3-44, reads as follows:

> The applicant indicates that Nakano fails to disclose or suggest the interface implemented on a plurality of computer platforms; however, Nakano is considered to provide for the feature to enable compatibility between units interconnected on a system bus, see col. 2 lines 39-49, by providing a consistent interaction interface regardless of the user's applications, see col. 1, lines 45-57, by dynamically linking load modules at runtime, col. 1 lines 60-68. further proof of the feature is specified via col. 2 lines 60-63, col. 3 lines 4-11, col. 11 lines 60-col. 12 line 9 and col. 20 lines 21-26.

1. First Point of Examiner's First Paragraph

The Applicant responds by addressing the first point of the Examiner's first paragraph, which reads as follows:

> The applicant indicates that Nakano fails to disclose or suggest the interface implemented on a plurality of computer platforms; however, Nakano is considered to provide for the feature to enable compatibility between units interconnected on a system bus, see col. 2 lines 39-49,

Nakano at col. 2 lines 39-49 cited by the Examiner, reads in part, as follows:

> A preferred embodiment of the invention is preferably practiced in the context of an operating system resident on a personal computer such as the IBM PS/2 or Apple Macintosh computer. A representative hardware environment is depicted in FIG. 1, which illustrates a typical hardware configuration of a workstation in accordance with the subject preferred embodiment having a central processing unit 10, such as a conventional microprocessor, and a number of other units interconnected via a system bus 12.

The Applicant responds that the microprocessor and other units interconnected by a system bus 12 in Nakano's Fig. 1 constitute a single personal computer system, as Nakano describes at col. 2, lines 11-13: "FIG. 1 is a block diagram of a personal computer system in accordance with a preferred embodiment of the subject invention". Nakano states that the work station of Fig. 1 has a single operating system, at col. 2, lines 60-64: "The workstation typically has resident thereon an operating system such as the IBM OS/2 operating system or the Apple System/7 operating system."

The Applicant respectfully avers that the Examiner is missing the basic point of the Applicants' claimed invention. The Applicants' claimed invention is about allowing the same application to run on different operating system platforms, because the Applicants have invented a common object-oriented interface implemented for each platform.

In contrast, Nakano's disclosed invention can be implemented on a plurality of platforms (Nakano/2:39-49) such as OS/2 and System 7 (Nakano/2:60-63), but Nakano is directed to a single operating system platform in each instance, upon which he builds an application with multiple languages or development environments (tool sets). Nakano's invention might then be practiced again on a different operating system platform to enable different applications to be built with different multiple development environments. However, the Applicants' claimed invention enables the same application to run on different operating system platforms, because the Applicants have invented a common object-oriented interface implemented for each platform.

The key thing the Examiner appears to be missing in the Applicants' claimed invention is the idea is that the same object-oriented application runs on the different combinations of operating system and hardware, because of the object-oriented interface implemented for each combination. There's no mention in Nakano of the same application running on the different operating systems, as is claimed by the Applicants.

Nakano fails to disclose or suggest the Applicants' claimed "interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems."

There is no disclosure or suggestion of the Applicants' claimed interface that is used by the same object-oriented application on the plurality of computer platforms to instantiate objects from the classes and invoke the object-oriented methods. There is no disclosure or suggestion of the recited program logic code on any one of the plurality of computer platforms that is responsive to the object-oriented interface implemented on the one computer platform to provide native operating system services from the one computer platform.

2. Second Point of Examiner's First Paragraph

The Applicant responds by addressing the second point of the Examiner's first paragraph, which reads as follows:

by providing a consistent interaction interface regardless of the user's applications, see col. 1, lines 45-57

Nakano at col. 1, lines 45-57 cited by the Examiner, reads in part, as follows:

> Object oriented applications should also reflect a consistent interaction interface with the user regardless of what application is currently active, and how many concurrent users are using the application. None of the prior art references applicant is aware of provides the innovative hardware and software system features which enable all object oriented applications to function in a consistent manner.
>
> Various development environments and tools produce load modules with a wide variety of formats and layouts. None of these environments use object oriented techniques to allow load modules to integrate as a single application.

The Applicant disagrees. The Examiner cites Nakano/1:45-57 about multiple applications having a "consistent interaction interface", but that is a statement about consistency across multiple applications running on the same operating system platform. The Examiner cites Nakano/3:4-11 regarding "different programming languages and development environments", but these are still on the same operating system platform. "Development environments" are not multiple operating systems each being its own development environment. On the contrary, Nakano is talking about different programming language tool sets (compilers, linkers, debuggers, etc.) running on the same operating system.

The Examiner states in the boldface paragraph 2: "native services (by different languages and different development environments – i.e. hardware systems)". But, the different language environments of Nakano are not on different "hardware systems," but rather they are on the same platform, for example, C++ and Lisp environments both running on the same Windows/Intel platform. Likewise, the different development environments of Nakano are not on different "hardware systems," but rather are on the same platform, for example, Borland, IBM, and Microsoft C++ language development environments all running on the same Windows/Intel platform.

Nakano fails to disclose or suggest the Applicants' claimed "interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems."

There is no disclosure or suggestion of the Applicants' claimed interface that is used by the same object-oriented application on the plurality of computer platforms to instantiate objects from the classes and invoke the object-oriented methods. There is no disclosure or suggestion of the recited program logic code on any one of the plurality of computer platforms that is responsive to the object-oriented interface implemented on the one computer platform to provide native operating system services from the one computer platform.

3. Third Point of Examiner's First Paragraph

The Applicant responds by addressing the third point of the Examiner's first paragraph, which reads as follows:

> by dynamically linking load modules at runtime, col. 1 lines 60-68. further proof of the feature is specified via col. 2 lines 60-63, col. 3 lines 4-11, col. 11 lines 60- col. 12 line 9 and col. 20 lines 21-26.

Nakano at col. 1 lines 60-68 cited by the Examiner, reads in part, as follows:

> The subject invention overcomes the deficiencies of the prior art by providing a system and method for dynamically linking load modules at runtime. A preferred embodiment employs a collection of load modules with different physical formats, each of which contains functions, static data, and classes to make the load modules appear as if they were statically linked together.

Nakano's col. 1 lines 60-68 referred to by the Examiner, recites "load module formats". Nakano describes "load modules" as application programs at col. 1, lines 53-57, which reads as follows:

> Various development environments and tools produce load modules with a wide variety of formats and layouts. None of these environments use object oriented techniques to allow **load modules to integrate as a single application**.

The "load module formats" referred to by the Examiner are Nakano's various versions of application programs, not the "program logic code" recited in Applicants' claims, which is "specific to the operating system and compiled for use on the computer hardware." Nakano is describing a first application making a call to a second application, not the Applicants' claimed "program logic code responsive to the object-oriented interface to provide native operating system services from the computer platform."

Nakano at col. 2 lines 60-63 cited by the Examiner, reads in part, as follows:

> "The workstation typically has resident thereon an operating system such as the IBM OS/2 operating system or the Apple System/7 operating system."

Nakano states that the single work station of Fig. 1 has a single operating system, at col. 2, lines 60-64. Nakano fails to disclose or suggest the Applicants' claimed "interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems."

Nakano at col. 3 lines 4-11 cited by the Examiner, reads in part, as follows:

> Similarly, different programming languages and development environments can use pre-existing TLoadModule subclasses, or design custom subclasses, to suit special needs. The TLoadModule class provides a common protocol for the loader to mix and match fundamentally different kinds of containers of data built by diverse development environments.

This cited disclosure in Nakano describes running applications with different languages concurrently in one computer. There is no disclosure or suggestion of the Applicants' claimed "interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems."

The Examiner has confused "development environments" with "hardware systems." Nakano defines "development environments" at col. 1, lines 53-57, which reads as follows:

> Various development environments and tools produce load modules with a wide variety of formats and layouts. None of these environments use object oriented techniques to allow load modules to integrate as a single application.

The Nakano disclosure of "development environments" at column 11, lines 25-35 shows that they are distinct from hardware. Column 11, lines 25-35 reads as follows:

> "Within the same CPU architecture, load modules created by different languages and by different development environments mix-and-match."

The Applicants' claimed invention is about same application running across multiple CPU architectures.

There is no disclosure or suggestion of the Applicants' claimed "interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems."

Nakano at col. 11 lines 60-col. 12 line 9 cited by the Examiner, reads in part, as follows:

> For example, suppose a subclass of TLoadModule is created from an environment other than C++, and any shared library (e.g., RunTime, LowLevelToolbox). Given a sufficiently robust development environment and the respective declarations used to build the original shared library, it is possible to reimplement the shared library in the other development environment. (Of course, some development environments may choose to restrict the ability of their classes to be subclassed by other environments; such an environment is not "sufficiently robust" according to this definition.) This thought experiment highlights the importance of each CPU architecture to define a common object layout, with respect to casting, virtual function calling, non-virtual function calling, and accessing public data members.

The Applicant would like to point out that the beginning of section at col. 11, lines 25-27, shows that the paragraph at col. 11 lines 60-col. 12 line 9 cited by the Examiner describes an example of Cross-Language Operability within a single CPU architecture. Col. 11, lines 25-27 reads as follows:

> Within the same CPU architecture, load modules created by different languages and by different development environments mix-and-match.

The Examiner cites Nakano/12:9 about "the importance of each CPU architecture to define a common object layout", but Nakano doesn't mean common object layout across multiple CPU architectures, but rather a common object layout across multiple development environments within a single given CPU architecture (a different CPU architecture would then have a new, different common object layout for itself).

There is no disclosure or suggestion of the Applicants' claimed "interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems."

Nakano at col. 20 lines 21-26 cited by the Examiner, reads in part, as follows:

> While the invention has been described in terms of a preferred embodiment in a specific system environment, those skilled in the art recognize that the invention can be practiced, with modification, in other and different hardware and software environments within the spirit and scope of the appended claims.

The Examiner is referring to Nakano's savings clause located at the very end of Nakano's specification, not a disclosure of Nakano's invention. This paragraph merely means practicing

the whole invention disclosed by Nakano over again independently on each of the several different hardware/software environments. Nakano does not address the problem of running the same application without modification on multiple hardware/operating system environments.

Nakano fails to disclose or suggest the Applicants' claimed object-oriented interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems, as discussed above.

There is no disclosure or suggestion of the Applicants' claimed interface that is used by the same object-oriented application on the plurality of computer platforms to instantiate objects from the classes and invoke the object-oriented methods. There is no disclosure or suggestion of the recited program logic code on any one of the plurality of computer platforms that is responsive to the object-oriented interface implemented on the one computer platform to provide native operating system services from the one computer platform.

B. The Examiner's second boldface paragraph:

The Examiner's second boldface paragraph: Office Action, Page 4, reads as follows:

> The applicant also indicates that Nakano does not disclose or suggest program logic code responsive to object oriented interface to provide native system services from the computer platform (i.e. a single platform not a plurality of platforms). However, see col. 20 lines 18-50 in which call are made between different formats (program code logic). Therefore, native services (by different languages and different development environments – i.e. hardware systems) are provide for as well as compatibility (Cross Language Compatibility) between the existing format (program logic code), see col. 11 lines 25-35.

1. First Sentence of Examiner's Second Paragraph

The Applicant responds by addressing the first sentence of the Examiner's second paragraph, which reads as follows:

> The applicant also indicates that Nakano does not disclose or suggest program logic code responsive to object oriented interface to provide native system services from the computer platform (i.e. a single platform not a plurality of platforms).

The Examiner is referring here that the Applicants' claim language which, for example in claim 10, reads:

"the program logic code responsive to the object-oriented interface to provide native system services from the computer platform;"

The Examiner is saying that Applicants' claim language recites a single platform and not to a plurality of platforms.

The Applicant disagrees. The Applicant has clearly claimed in claim 10 that the object-oriented "interface is implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems". Moreover, the Applicant has clearly recited in the claim the "computer platform including computer hardware and an operating system executing on the computer hardware, including program logic code specific to the operating system and compiled for use on the computer hardware."

Thus, it is clear from the language in the Applicants' claims, that when the object-oriented interface is implemented on a first type computer platform which has a first-type operating system and a first type computer hardware, that a first type program logic code that is specific to the first type operating system and compiled for use on the first type computer hardware of the first type platform, will be responsive to the object-oriented interface implemented on the first type computer platform. The interface is used by the same object-oriented application on each of a plurality of computer platforms to instantiate objects from the classes and invoke the object-oriented methods. The program logic code on the first computer platform is responsive to the object-oriented interface implemented on the first computer platform to provide native operating system services from the first computer platform.

Similarly, it is clear from the language in the Applicants' claims, that when the object-oriented interface is implemented on a second type computer platform which has a second-type operating system and a second type computer hardware, that a second type program logic code that is specific to the second type operating system and compiled for use on the second type computer hardware of the second type platform, will be responsive to the object-oriented interface implemented on the second type computer platform. The interface is used by the same object-oriented application on the second computer platform to instantiate objects from the classes and invoke the object-oriented methods. The program logic code on the second computer platform is responsive to the object-oriented interface implemented on the second computer platform to provide native operating system services from the second computer platform.

Nakano fails to disclose or suggest the Applicants' claimed "the program logic code responsive to the object-oriented interface to provide native operating system services from the computer platform." Native system services are services provided by an underlying operating system, which are functionalities that cannot otherwise be readily implemented in an application program including services such as tasks, virtual memory, interprocess communications (IPC), synchronization, scheduling, fault, machine, and security services, as shown in Applicants' Figure 4.

Nowhere in Nakano does the term "services" appear, nor does any teaching or suggestion that the code loaded by Nakano is for enabling an application to access operating system services such as those enumerated in Applicants' Fig. 4 that are provided by and native to the underlying computer platform (operating system and computer hardware) in the Applicants' claimed invention.

As to the issue of native system services, the Examiner is citing Nakano's various library linking and loading functions as examples of native system services. Those are not "native operating system services". Instead, "native operating system services" are the classic "system services" of an operating system, related to managing computer (hardware) resources, e.g. threads, memory, file system, networking, display.

Nakano fails to disclose or suggest the Applicants' claimed "the program logic code responsive to the object-oriented interface to provide native operating system services from the computer platform."

There is no disclosure or suggestion of the Applicants' claimed interface that is used by the same object-oriented application on the plurality of computer platforms to instantiate objects from the classes and invoke the object-oriented methods. There is no disclosure or suggestion of the recited program logic code on any one of the plurality of computer platforms that is responsive to the object-oriented interface implemented on the one computer platform to provide native operating system services from the one computer platform.

2. Second Sentence of Examiner's Second Paragraph

The Applicant responds by addressing the Examiner's second sentence of the second paragraph, which reads as follows:

> However, see col. 20 lines 18-50 in which call are made between different formats (program code logic).

The Applicant disagrees. Nakano at col. 20 lines 18-50 cited by the Examiner, reads further, in part, as follows:

> 1. An apparatus for processing a plurality of load modules in a computer, said plurality of load modules created by a plurality of development environments resulting in different load module formats, comprising:
> (a) the computer;
> (b) a storage in the computer;
> (c) processing means in the computer for loading a first object-oriented load module of a first format in said address space;
> (d) processing means in the computer for loading a second load module which can be non-object-oriented of a second format in said address space; and
> (e) processing means in the computer for having said first object-oriented load module and said second load module execute in said address space and utilize portions of said second load module for execution of said first object-oriented load module.
> 2. An apparatus for processing a plurality of load modules in an address space as recited in claim 1, wherein the processing means in element (e) include means for said first object-oriented load module to call a function in said second load module.

The Examiner is referring to Nakano's claims 1 and 2. The "call made between different formats" is the Examiner's characterization of Nakano's claims' recitation of "first object-oriented load module [of a first format] to call a function in said second load module [of a second format]."

Nakano's claims 1 and 2 referred to by the Examiner, recite "load module formats". Nakano describes "load modules" as application programs at col. 1, lines 53-57, which reads as follows:

> Various development environments and tools produce load modules with a wide variety of formats and layouts. None of these environments use object oriented techniques to allow **load modules to integrate as a single application**.

The "load module formats" referred to by the Examiner are Nakano's various versions of application programs, not the "program logic code" recited in Applicants' claims, which is "specific to the operating system and compiled for use on the computer hardware." Nakano is

describing a first application making a call to a second application, not the Applicants' claimed "program logic code responsive to the object-oriented interface to provide native operating system services from the computer platform."

The Examiner further referred to Nakano at col. 20 lines 18-50, which reads in part, as follows:

> While the invention has been described in terms of a preferred embodiment in a specific system environment, those skilled in the art recognize that the invention can be practiced, with modification, in other and different hardware and software environments within the spirit and scope of the appended claims.

The Examiner is referring to Nakano's savings clause located at the very end of Nakano's specification, not a disclosure of Nakano's invention. This paragraph merely means practicing the whole invention disclosed by Nakano over again independently on each of the several different hardware/software environments. Nakano does not address the problem of running the same application without modification on multiple hardware/operating system environments. The savings clause is not a disclosure or suggestion of the Applicants' claimed invention. Nakano fails to disclose or suggest the Applicants' claimed object-oriented interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems, as discussed above.

There is no disclosure or suggestion of the Applicants' claimed interface that is used by the same object-oriented application on the plurality of computer platforms to instantiate objects from the classes and invoke the object-oriented methods. There is no disclosure or suggestion of the recited program logic code on any one of the plurality of computer platforms that is responsive to the object-oriented interface implemented on the one computer platform to provide native operating system services from the one computer platform.

3. Third Sentence of Examiner's Second Paragraph

The Applicant responds by addressing the third sentence of the second paragraph, which reads as follows:

> Therefore, native services (by different languages and different development environments – i.e. hardware systems) are provide for as well as compatibility (Cross Language Compatibility) between the existing format (program logic code), see col. 11 lines 25-35.

The Examiner has confused "development environments" with "hardware systems." Nakano defines "development environments" at col. 1, lines 53-57, which reads as follows:

> Various development environments and tools produce load modules with a wide variety of formats and layouts. None of these environments use object oriented techniques to allow load modules to integrate as a single application.

Nakano's column 11, lines 25-35 cited by the Examiner, read as follows:

> Within the same CPU architecture, load modules created by different languages and by different development environments mix-and-match. The preferred embodiment of the preferred embodiment requires that addresses corresponding to exported functions and static data items follow a specific CPU's calling convention, which may involve providing addresses that correspond to translating between the preferred embodiment of the subject preferred embodiment's calling conventions, and a different internal set of conventions.

The Applicants reply that Nakano fails to disclose or suggest the Applicants' claimed "interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems." The Nakano disclosure at column 11 reads: "Within the same CPU architecture, load modules created by different languages and by different development environments mix-and-match." This cited disclosure in Nakano describes running applications with different languages concurrently in one computer. There is no disclosure or suggestion of the Applicants' claimed "interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems."

Nakano fails to disclose or suggest the Applicants' claimed "the program logic code responsive to the object-oriented interface to provide native system services from the computer platform." Native system services are services provided by an underlying operating system which are functionalities that cannot otherwise be readily implemented in an application program including services such as tasks, virtual memory, interprocess communications (IPC), synchronization, scheduling, fault, machine, and security services, as shown in Applicants' Figure 4.

Nowhere in Nakano does the term "services" appear, nor does any teaching or suggestion that the code loaded by Nakano is for enabling an application to access system services such as those enumerated in Applicants' Fig. 4 that are provided by and native to the underlying

computer platform (operating system and computer hardware) in the Applicants' claimed invention.

There is no disclosure or suggestion of the Applicants' claimed interface that is used by the same object-oriented application on the plurality of computer platforms to instantiate objects from the classes and invoke the object-oriented methods. There is no disclosure or suggestion of the recited program logic code on any one of the plurality of computer platforms that is responsive to the object-oriented interface implemented on the one computer platform to provide native operating system services from the one computer platform.

C. The Examiner's third boldface paragraph:

The Examiner's third boldface paragraph: Office Action, Page 4, reads as follows:

> Various native services are provided to enable compatibility, see Nakano's section entitled Cross Language Compatibility in col. 11-col. 12. Services also are considered to enable dynamic linking at runtime to features of different languages and different environments, see the abstract and col. 11 lines 25-35. therefore, each of the features are considered provided for.

Nakano at col. 11-col. 12 cited by the Examiner, reads further, in part, as follows:

> Cross-language operability
>
> Within the same CPU architecture, load modules created by different languages and by different development environments mix-and-match. The preferred embodiment of the preferred embodiment requires that addresses corresponding to exported functions and static data items follow a specific CPU's calling convention, which may involve providing addresses that correspond to translating between the preferred embodiment of the subject preferred embodiment's calling conventions, and a different internal set of conventions. For instance, a load module created by a non-native C++ environment (e.g. Lisp) can mix-and-match freely if:
>
> 1) a call into the load module to a function address of an exported function points to code consistent with calling conventions,
>
> 2) a call out of the load module goes through glue code that converts from the internal conventions to conventions corresponding to the preferred embodiment of the subject preferred embodiment,
>
> 3) a read to a static data address for an exported static data item causes the correct value to appear in some specified destination,

4) a write to a static data address for an exported static data item causes the data item to obtain the correct value from some specified source,

5) casting to a more derived class and casting to a base class both work, even if the classes in the hierarchy aren't all defined by the same language,

6) accessing a public data member or a class static data member, if any, works, and

7) a virtual function call to the TClassHandle or TFunctionHandle object for an exported class, function, or static data item, respectively, follows the protocol for the class.

For example, suppose a subclass of TLoadModule is created from an environment other than C++, and any shared library (e.g., RunTime, LowLevelToolbox). Given a sufficiently robust development environment and the respective declarations used to build the original shared library, it is possible to reimplement the shared library in the other development environment. (Of course, some development environments may choose to restrict the ability of their classes to be subclassed by other environments; such an environment is not "sufficiently robust" according to this definition.) This thought experiment highlights the importance of each CPU architecture to define a common object layout, with respect to casting, virtual function calling, non-virtual function calling, and accessing public data members.

The Examiner is citing Nakano's various library linking and loading functions as examples of native system services. Those are not "native operating system services". Instead, "native operating system services" are the classic "system services" of an operating system, related to managing computer (hardware) resources, e.g. threads, memory, file system, networking, display.

Nakano fails to disclose or suggest the Applicants' claimed "the program logic code responsive to the object-oriented interface to provide native operating system services from the computer platform."

Native system services are services provided by an underlying operating system which are functionalities that cannot otherwise be readily implemented in an application program including services such as tasks, virtual memory, interprocess communications (IPC), synchronization, scheduling, fault, machine, and security services, as shown in Applicants' Figure 4.

Nowhere in Nakano does the term "services" appear, nor does any teaching or suggestion that the code loaded by Nakano is for enabling an application to access system services such as those enumerated in Applicants' Fig. 4 that are provided by and native to the underlying computer platform (operating system and computer hardware) in the Applicants' claimed invention.

The Applicant would like to point out that the beginning of section at col. 11, lines 25-27, shows that the paragraph at col. 11 lines 60-col. 12 line 9 cited by the Examiner describes an example of Cross-Language Operability within a single CPU architecture. Col. 11, lines 25-27 reads as follows:

> Within the same CPU architecture, load modules created by different languages and by different development environments mix-and-match.

This cited disclosure in Nakano describes running applications with different languages concurrently in one computer. There is no disclosure or suggestion of the Applicants' claimed "interface implemented on a plurality of computer platforms including different combinations of computer hardware and operating systems."

There is no disclosure or suggestion of the Applicants' claimed interface that is used by the same object-oriented application on the plurality of computer platforms to instantiate objects from the classes and invoke the object-oriented methods. There is no disclosure or suggestion of the recited program logic code on any one of the plurality of computer platforms that is responsive to the object-oriented interface implemented on the one computer platform to provide native operating system services from the one computer platform.

CONCLUSION

Based on the foregoing remarks, Applicants respectfully request reconsideration and withdrawal of the rejection of claims and allowance of this application.

AUTHORIZATION

The Commissioner is hereby authorized to charge any additional fees which may be required for consideration of this Amendment to Deposit Account No. **13-4500**, Order No. 4386-7004US1.

In the event that an extension of time is required, or which may be required in addition to that requested in a petition for an extension of time, the Commissioner is requested to grant a petition for that extension of time which is required to make this response timely and is hereby authorized to charge any fee for such an extension of time or credit any overpayment for an extension of time to Deposit Account No **13-4500**, Order No. 4386-7004US1.

Respectfully submitted,
MORGAN & FINNEGAN, L.L.P.

Dated: October 31, 2007 By: _____

John E. Hoel
Registration No. 26,279
(202) 857-7887 Telephone
(202) 857-7929 Facsimile

Correspondence Address:
MORGAN & FINNEGAN, L.L.P.
3 World Financial Center
New York, NY 10281-2101